Progetto di Fondamenti di Intelligenza Artificiale

Kawaii-Bot: Sistema di Raccomandazione di Kawaii-Comix

Autore	**Matricola**
Davide Del Franco Natale	0512113233
Simone D'Assisi	0512113584

Link: Kawaii-Bot || Kawaii-Comix

Contents

1. Introduzione

Il modulo di Intelligenza Artificiale **Kawaii-Bot** per l'e-commerce *Kawaii-Comix* è stato progettato per migliorare l'esperienza di shopping per gli appassionati di manga. Grazie a Kawaii-Bot, i clienti possono accedere a suggerimenti personalizzati e pertinenti, basati sui loro gusti, preferenze di acquisto e sulle ultime tendenze del settore. Questo significa che i clienti non dovranno più passare ore a cercare prodotti, ma potranno ricevere raccomandazioni su misura per loro, rendendo il processo di acquisto più efficiente e soddisfacente.

Inoltre, l'implementazione di un sistema di raccomandazioni personalizzate può aiutare a migliorare l'engagement dei clienti, incoraggiandoli a esplorare nuovi prodotti e categorie che potrebbero aver ignorato in precedenza. Questo non solo migliora l'esperienza di shopping, ma può anche portare a un aumento delle vendite e della fidelizzazione dei clienti.

1.1 Metodologia Utilizzata

CRISP-DM è l'acronimo di Cross-Industry Standard Process for Data Mining, un metodo di comprovata efficacia per l'esecuzione di operazioni di data mining. Come metodologia, comprende descrizioni delle tipiche fasi di un progetto e delle attività incluse in ogni fase e fornisce una spiegazione delle relazioni esistenti tra tali attività.

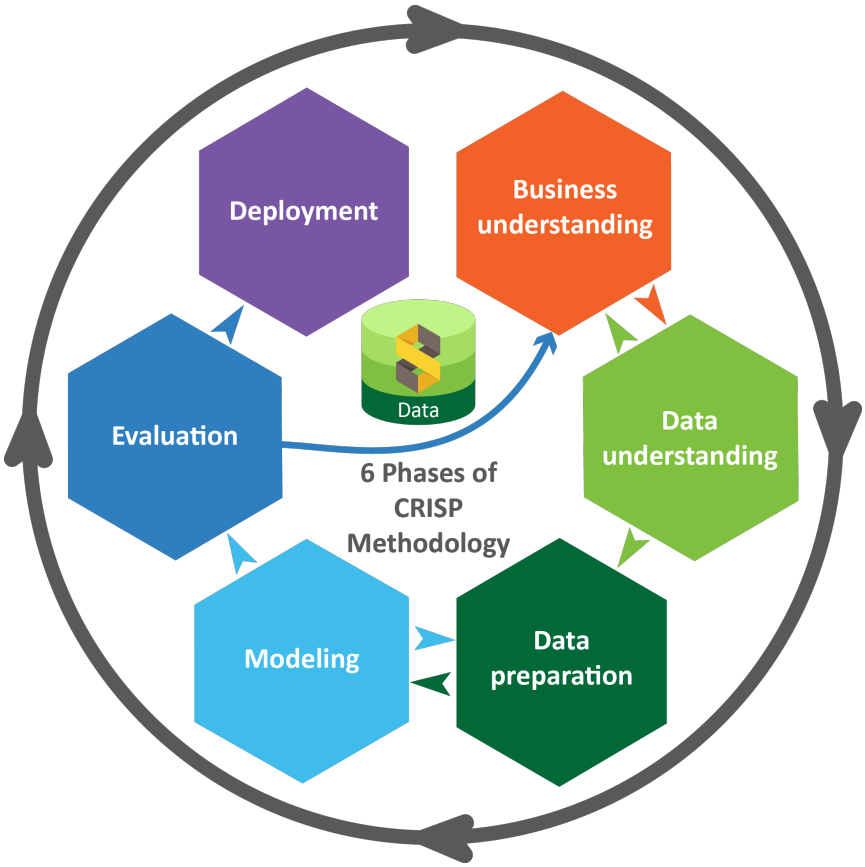


Figure 1.1: Fasi della Metodologia CRISP-DM

2. Business Understanding

2.1 Obiettivi

Lo scopo del progetto consiste nella realizzazione di un modulo di raccomandazione personalizzata per l'e-commerce *Kawaii-Comix* che usi un approccio di tipo **content-based**. Questo modulo dovrà essere in grado di suggerire all'utente diversi prodotti affini ai suoi gusti e alle sue abitudini d'acquisto, basandosi sul contenuto dei prodotti stessi. Gli obbiettivi specifici includono:

- migliorare l'esperienza dell'utente mostrando suggerimenti personalizzati che risultino simili al prodotto visualizzato;
- aumentare le interazioni degli utenti con l'e-commerce e in definitiva aumentare le vendite;
- fidelizzare i clienti offrendo un servizio di qualità personalizzato che soddisfi le loro esigenze e le loro aspettative.

2.2 Specifiche P.E.A.S

PEAS è l'acronimo inglese di Performance Environment Actuators Sensors. È utilizzato nello studio dell'intelligenza artificiale per raggruppare in un unico termine l'ambiente operativo.

Performance. la misura di prestazione considerata prevede l'avvicinamento il più possibile allo scenario in cui a ciascun utente verranno mostrati solamente i prodotti altamente rilevanti per loro e che potrebbero acquistare.

Environment. l'ambiente in cui opera l'agente è l'e-commerce Kawaii-Comix, dove ha la possibilità di accedere alle informazioni che riguardano i prodotti presenti nel catalogo e gli ordini effettuati dagli utenti. Il nostro ambiente risulta:

- **Completamente osservabile:** l'agente ha accesso a tutte le informazioni necessarie che riguardano sia utenti che prodotti in ogni momento;
- **Stocastico:** lo stato dell'ambiente cambia indipendentemente dalle decisioni prese dagli utenti;
- **Sequenziale:** le decisioni prese dagli utenti in precedenza influenzato le decisioni future che l'agente prenderà;
- **Dinamico:** l'ambiente potrebbe variare mentre viene fatta una previsione, in quanto un utente potrebbe effettuare un ordine, variando così le proprie preferenze;
- **Discreto:** le previsioni dipendono dalle caratteristiche dei prodotti considerati;
- **Ad Agente Singolo:** vi sarà un unico agente ad operare.

Actuators. l'agente agisce sull'ambiente presentando una lista di prodotti suggeriti che compariranno sulle pagine dei singoli prodotti durante la navigazione.

Sensors. l'agente percepisce le informazioni attraverso l'accesso diretto al Database del sito Kawaii-Comix.

2.3 Requisiti

Il sistema proposto dovrà essere in grado di:

- Raccogliere e analizzare i dati relativi al contenuto dei prodotti, in riferimento specialmente ai tag associati;
- Utilizzare algoritmi content-based per generare suggerimenti personalizzati in tempo reale, basati sul contenuto dei prodotti stessi;
- Integrarsi in modo efficiente con il sistema esistente di gestione dei contenuti e dei dati dell'e-commerce Kawaii-Comix;
- Garantire la privacy e la sicurezza dei dati degli utenti conformemente alle normative sulla privacy vigenti.

2.4 Risorse e Tool

Ci avvarremo del sito *kaggle*[1] per l'acquisizione del dataset, fondamentale per il nostro sistema di raccomandazione. Per l'analisi, la modellazione, l'addestramento e la visualizzazione grafica dei dati e del modello verranno utilizzati diversi tool, trai quali: Python in concomitanza di varie librerie, come pandas, numpy, matplotlib e seaborn per le informazioni sui dati e sklearn per la fase di feature engineering e la fase di modeling. Per l'integrazione con Kawaii-Comix si useranno Flask e Ajax.

[1] Kaggle è una piattaforma che consente agli utenti di trovare e pubblicare set di dati, esplorare e costruire modelli in un ambiente di data science basato sul web, collaborare con altri scienziati dei dati e ingegneri di machine learning e partecipare a competizioni per risolvere sfide di data science.

3. Data Understanding

In questa seconda fase del CRISP-DM verrà analizzato il datasaet in modo approfondito per comprendere la sua struttura, il contenuto, le relazioni tra le variabili e le eventuali problematiche o limitazioni dei dati.

3.1 Raccolta dei dati

Il dataset scelto è stato reperito da *kaggle* in formato *cvs* al seguente indirizzo. Le informazioni del dataset provengono da un **dump** effettuato sul database del sito web *Anime-Planet* [1] e contenente tutti i Manga, Manwha e Manhua presenti sul sito fino al 2022.

3.2 Descrizione dei dati

Il dataset considerato presenta 70.948 prodotti, tra Manga, Manwha e Manhua. Di ogni prodotto vengono riportate le seguenti caratteristiche:

- Titolo (title): nome completo del manga;
- Valutazione (rating): punteggio medio dato dagli utenti per ogni manga;
- Descrizione (description): breve descrizione del manga;
- Anno (year): anno di pubblicazione del manga;
- Tag (tags): una lista dei tag (generi e categorie) dei manga;
- Copertina (cover): URL della copertina del manga.

title	description	rating	year	tags	cover
The Master of Diabolism	As the grandmaster who founded the Demonic Sect...	4.7	2017	['Action', 'Adventure', ...]	https://cdn.anime-planet.com/...
...	...	...	...	...	...

Table 3.1: Esempio di un prodotto nel dataset

[1] **Anime-Planet.** sito web che include numerose informazioni riguado anime e manga. Comprende anche review e aspetti relativi ai social media, come forum e un canale Discord dedicato.

3.3 Esplorazione dei dati

In questa fase verranno condotte analisi esplorative per scoprire schemi, tendenze, correlazioni o anomalie nei dati. Tutte le analisi verranno condotte con l'ausilio di **Python**, in particolare utilizzando le librerie **pandas**[2] e **matplotlib**[3].

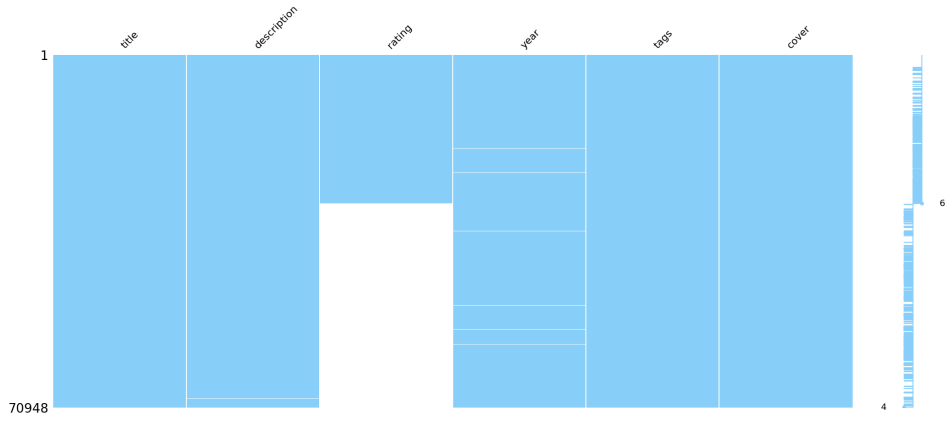
3.3.1 Valori *null*

Come primo passo, verrà esplorato il dataset alla ricerca di valori *null* attraverso il seguente codice Python:

```
#Conta e stampa i valori null per ogni attributo
nan_mask = manga_dataset.isna()
nan_count = nan_mask.sum()
print(nan_count)
```

Di seguito verranno riportati il numero di valori null per ogni attributo del dataset:

title	description	rating	year	tags	cover
0	31	41077	824	0	0



Possiamo notare che più della metà dei valori dell'attributo rating sono null.

3.3.2 Valori Duplicati

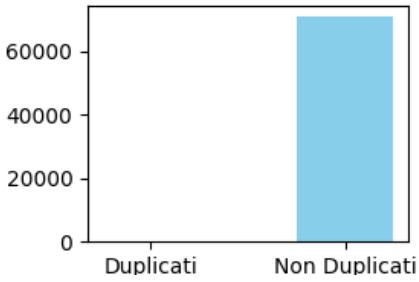
Si ricercano in questa fase i valori duplicati attraverso il seguente codice:

```
# Ricerca dei valori duplicati
duplicated_values = manga_dataset.duplicated().sum()
not_duplicated_values = (~manga_dataset.duplicated()).sum()
print("Valori duplicati: ", duplicated_values)
print("Valori non duplicati: ", not_duplicated_values)
```

Duplicati	Non Duplicati
0	70948

[2]pandas è un tool open source veloce, potente, flessibile e facile da usare per l'analisi e manipolazione dei dati, costruito per Python.

[3]Matplotlib è una libreria completa per creare visualizzazioni statiche, animate e interattive in Python.

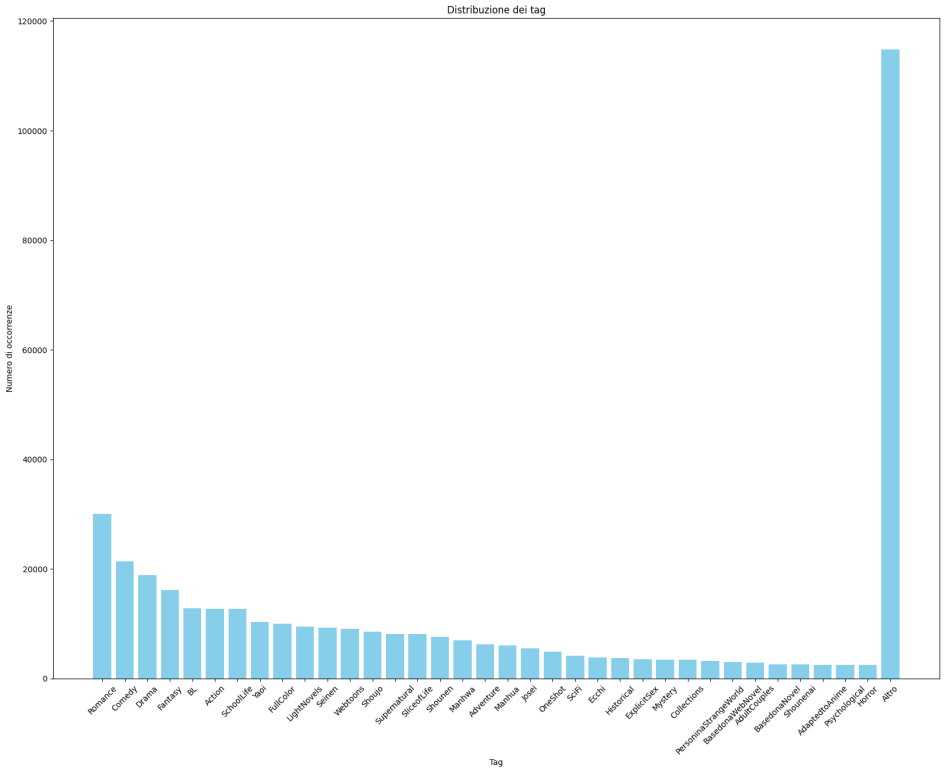


Possiamo affermare quindi che non ci siano duplicati nel dataset preso in esame.

3.3.3 Analisi della Distribuzione dei Tag

Vengono identificati i tag più comuni, così da comprendere quali sono i generi e le categorie più popolari tra gli utenti e influenzare di conseguenza la raccomandazione. Il codice utilizzato per l'analisi della distribuzione è il seguente:

```
    # Conta le ripetizioni di ogni tag
    tag_counts = Counter(all_tags_list)

    # Estrai i top_n tag più comuni e i loro conteggi, dove top_n=35
    top_tags = tag_counts.most_common(top_n)
    other_count = sum(tag_counts.values()) - sum(count for tag, count in top_tags)

    # Seleziona i tag e i conteggi per il grafico
    tags = [tag for tag, count in top_tags] + ['Altro']
    counts = [count for tag, count in top_tags] + [other_count]
```



3.3.4 Analisi dei Valori di Rating

Analisi descrittiva

Verranno qui osservati alcuni valori che riguardano l'attributo *Rating* (ove presente):

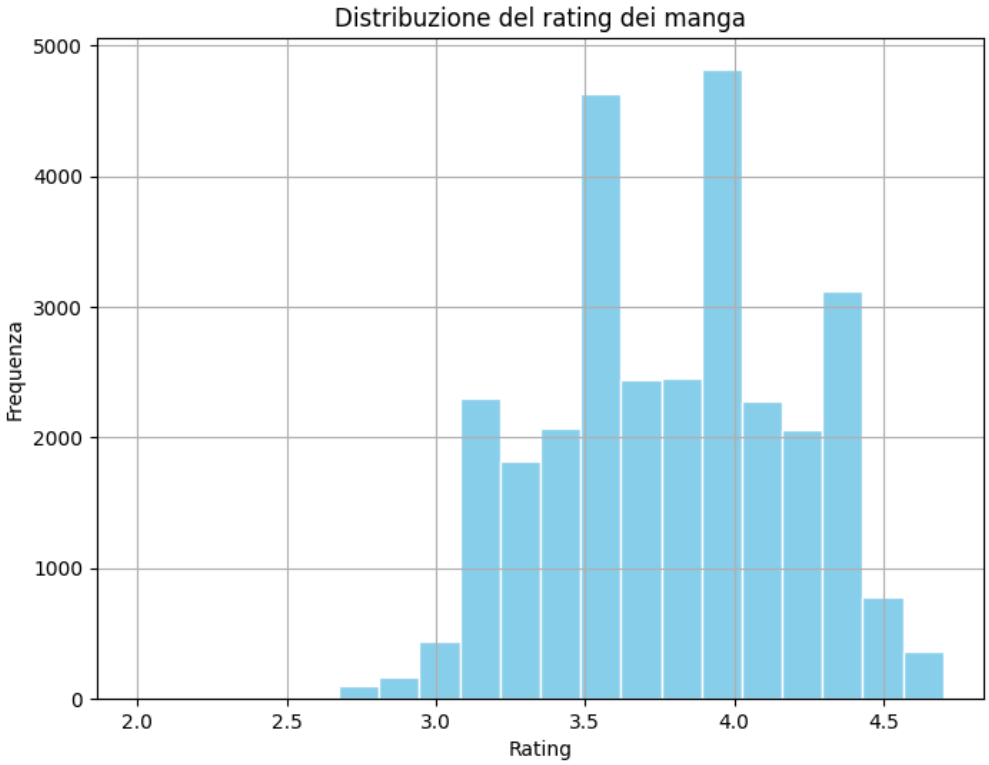
- **Totale:** Il numero totale di valori presenti nel dataset per l'attributo Rating. In questo caso, ci sono **29.871** valori di Rating;
- **Media:** La media dei valori di Rating. In questo caso, la media del Rating è di circa **3.78**;
- **Deviazione Std:** La deviazione standard dei valori di Rating. Misura la dispersione dei valori di Rating intorno alla media. In questo caso, la deviazione standard è di circa **0.40**, il che significa che i valori di Rating tendono a variare di circa 0.40 dalla media;
- **Minimo:** Il valore minimo di Rating nel dataset. In questo caso, il rating minimo è di **2.0**;
- **Massimo:** il valore massimo di Rating nel dataset. In questo caso, il rating massimo è di **4.7**.

Totale	Media	Deviazione Std	Minimo	Massimo
29871	3.78	0.4	2	4.7

Table 3.2: Tabella Riassuntiva

Distribuzione del Rating

Come si può notare dall'immagine 3.3.4 la distribuzione risulta asimmetrica: ha due picchi distinti, uno intorno al valore 3.5 e l'altro intorno al valore 4.0. Poiché la distribuzione non è simmetrica e regolare, allora possiamo dire che essa non è normale.

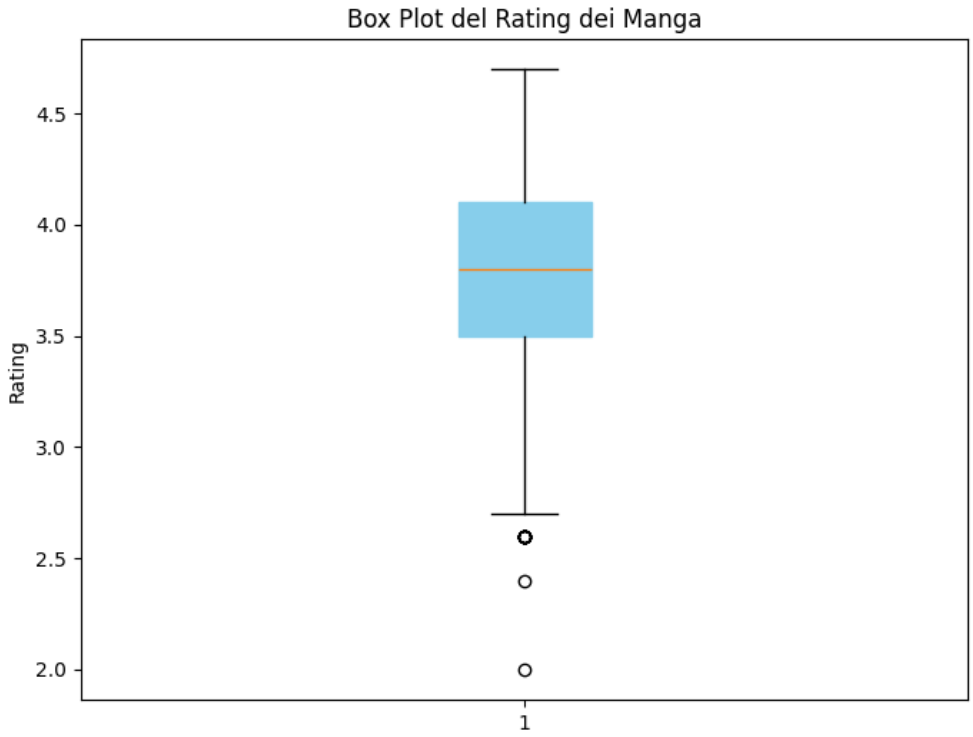


Ricerca Outlier

Poiché è emerso che la distribuzione del Rating non è normale, vale la pena verificare se sono presenti *outlier*[4] nei dati. Gli outlier possono influenzare significativamente la stima dei parametri del modello in quanto influenzano in modo diretto media e varianza. Per trovare gli outlier verrà usato il metodo dell'**Intervallo Interquartile** attraverso il seguente codice Python:

```
rating = mangadataset['rating']

# Calcola il primo e il terzo quartile
Q1 = rating.quantile(0.25)
Q3 = rating.quantile(0.75)

# Calcola l'intervallo interquartile (IQR)
IQR = Q3 - Q1

# Calcola i limiti per identificare gli outlier
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Trova gli outlier
outlier_indices = rating[(rating < lower_bound) | (rating > upper_bound)].index.
tolist()

outlier_ratings = mangadataset.loc[outlier_indices, 'rating']
print(outlier_ratings)
```

Applicando il codice al nostro dataset, emerge che esistono 14 outlier che, come evidenziato dalla figura 4.2 hanno una valutazione pari a 2.6 o inferiore.



[4]Outlier è un termine utilizzato in statistica per definire, in un insieme di osservazioni, un valore anomalo e aberrante, ossia un valore chiaramente distante dalle altre osservazioni disponibili

3.4 Verifica della qualità dei dati

Dalle analisi condotte nella fase di Data Understanding sono emerse diverse problematiche che verranno trattate nella fase di Data Preparation:

1. **Valori Null:** come emerso durante l'esplorazione dei dati, in riferimento alla Sezione 3.3.1, sono presenti 41077 valori null per l'attributo *rating*, 31 valori per l'attributo *description* e 824 per l'attributo *year*. Questi valori null potrebbero influenzare l'analisi successiva e saranno quindi gestiti durante la prossima fase.

2. **Valori Duplicati:** durante l'esplorazione è stato confermato che non sono presenti valori duplicati nel dataset, come evidenziato dalla Sezione 3.3.2. Questo è un risultato positivo in quanto non sarà necessario gestirli nella fase di Data Preparation. La mancanza di valori duplicati contribuisce alla pulizia e all'affidabilità del dataset.

3. **Analisi della Distribuzione dei Tag:** nella Sezione 3.3.3 è stata effettuata un'analisi approfondita per quanto riguarda la distribuzione dei tag più comuni, trai quali vengono ad esempio identificati "Romance", "Comedy" o "Drama". Questo ci aiuta a comprendere quali siano i tag più popolari tra gli utenti, informazione cruciale per il sistema di raccomandazione *Kawaii-Bot* poichè andrà a influenzare la fase successiva dell'analisi.

4. **Analisi dei Valori di Rating:** nella sezione 3.3.4 è emerso che la distribuzione dei valori di Rating non è normale, in quanto abbiamo due picchi per i valori di 3.5 e 4.0. Questo ci suggerisce che questi valori sono comuni per i manga valutati. Inoltre sono emersi 14 outlier tra le entry con una valutazione pari a 2.6 o inferiore.

4. Data Preparation

Dopo aver acquisito e analizzato i dati, nella fase di Data Preparation questi vengono preparati affinché possano essere utilizzati in fase di addestramento dell'algoritmo di Machine Learning. In questa fase risulta utile l'uso della libreria **sklearn**[1], specialmente per il processo di Feature Engineering.

4.1 Data Cleaning

Sulla base delle informazioni ottenute durante la fase di Data Understanding, verrà qui deciso come risolvere i problemi relativi alla qualità dei dati:

- La maggior parte dei valori *null* sono concentrati nella colonna relativa al "rating" dei prodotti. Tuttavia l'attributo rating non verrà utilizzato dalla versione attuale del modello da noi proposto. Si rimanda alla sezione 7.2 per ulteriori considerazioni.
- La mancata considerazione della colonna di rating risolve anche il problema relativo alla presenza di outlier.
- Tra gli attributi presenti nel dataset preso in esame, l'*anno* non verrà considerato in quanto il sito di riferimento Kawaii-Comix non ne tiene traccia. Inoltre, anche la *copertina* e la *descrizione* non saranno considerate in quanto irrilevanti ai fini del nostro sistema.

Pulizia dei Tag

Per utilizzare i tag in modo efficace, è stato inoltre ritenuto necessario pulirli, andando a rimuovere caratteri speciali o spazi che avrebbero potuto confondere il modello. Per la pulizia sono state usate le seguenti funzioni:

```
    # Elimina gli spazi presenti all'interno del singolo tag
    return match.group(0).replace(" ", "")
```

```
    # Converti il testo in minuscolo
    text = text.lower()

    # Rimuovi i numeri
    text = re.sub(r'\d+', '', text)

    # Rimuovi i caratteri speciali e simboli (mantieni la virgola e rimuovi il singolo apice)
    text = re.sub(r"[^a-zA-Z\s,]", '', text)

    # Rimuovi gli spazi extra
    text = ' '.join(text.split())

    return text
```

Partendo ad esempio da una lista di tag del tipo *['Action', 'Shonen ai', 'Based on a Novel']* si otterrà una lista di tag del tipo *action, shonenai, basedonanovel*.

[1] Libreria che offre strumenti semplici ed efficienti per l'analisi predittiva dei dati, accessibili a tutti e riutilizzabili in diversi contesti, sviluppati su NumPy, SciPy e matplotlib, open source e utilizzabili commercialmente con licenza BSD.

4.2 Feature Engineering

Poiché l'obbiettivo del nostro progetto è consigliare i prodotti da acquistare in base alla loro similarità, abbiamo deciso di concentrarci sull'attributo *tag*. Tuttavia il nostro modello di Machine Learning non può prendere in input valori non numerici, ed essendo i tag delle stringhe di testo è necessario trovare un modo per codificarli in modo efficace affinchè possano essere presi in input.

One-Hot Encoding

Utilizzeremo la **codifica One-Hot** per la rappresentazione dei tag secondo questi passaggi:

1. Identificazione Categorie Uniche: vengono inizialmente identificate tutte le categorie uniche presenti nella variabile categorica *Tags* a seguito della pulizia, come ad esempio "romance", "drama" o "comedy".

2. Creazione Variabili Binarie: per ogni categoria unica individuata, viene creata una nuova variabile binaria, detta **indicatore**.

3. Assegnazione Indicatori: per ogni riga dei dati, gli indicatori vengono impostati a 1 se la categoria corrispondente è presente, altrimenti vengono impostati a 0.

Utilizzeremo il **CountVectorizer** per applicare la codifica One-Hot ai tag. Infatti, il CountVectorizer converte una collezione di documenti in un vettore di conteggio delle parole, ponendo attenzione alla frequenza con cui ogni parola compare. Un limite del CountVectorizer è che attribuisce anche ad articoli, congiunzioni, preposizioni la stessa importanza di altre parole che invece dovrebbero avere più peso, come i sostantivi. Tuttavia risulta adatto per la costruzione del sistema di raccomandazione da noi proposto, in quanto non andrà a lavorare con frasi intere ma invece con i tag. Avremo quindi a che fare con parole che avranno la stessa importanza. Per applicare la codifica One-Hot con CountVectorizer al nostro dataset viene utilizzata la libreria **sklearn**, come mostrato nel seguente codice:

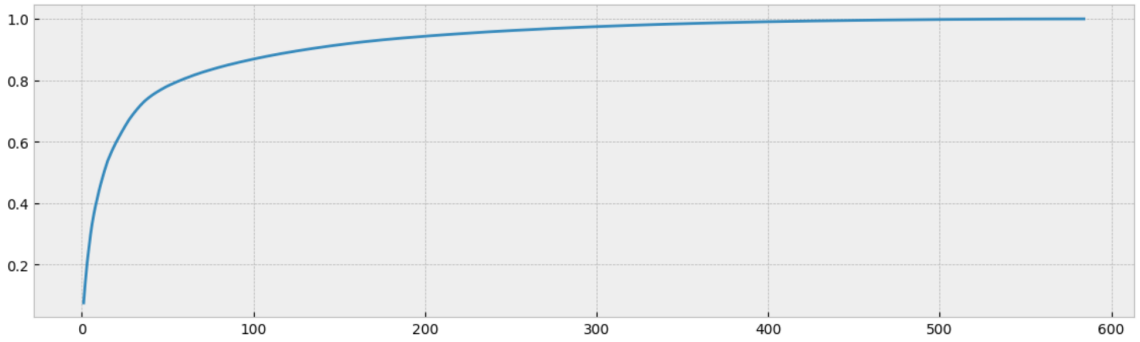
```
vectorizer = CountVectorizer()
# Codifica ogni prodotto con il CountVectorizer
labels = vectorizer.fit_transform(df['tags'])
# Crea un nuovo dataframe
X = pd.DataFrame(labels.toarray(), columns=vectorizer.get_feature_names_out())
```

Al termine della codifica One-Hot risulta che le colonne individuate nel nuovo dataset sono **583**, una per ogni categoria unica.

Principal Component Analysis (PCA)

La **Principal Component Analysis**, o PCA, è una tecnica di analisi dei dati utilizzata per ridurre la dimensionalità del dataset mantenendo la maggior parte delle informazioni contenute nei dati originali. Partendo dalle 583 feature considerate ci siamo occupati di valutare quanto queste feature potessero essere ridotte senza una significativa perdita di informazione.

Varianza Spiegata Cumulativa. Per il calcolo del PCA risulta di fondamentale importanza considerare la varianza spiegata cumulativa: essa misura quanto ciascuna feature contribuisce a catturare una determinata quantità di informazione dal dataset. Il grafico della varianza cumulativa mostra quanto della variazione totale nei dati può essere catturata utilizzando un numero crescente di componenti principali. Utilizzeremo la libreria *matplotlib* per mostrare il grafico della varianza spiegata cumulativa:



Si è notato che è possibile ridurre notevolmente il numero di feature mantenendo il 95% dell'informazione. Per la precisione, attraverso il codice seguente abbiamo stimato con esattezza che il numero di feature da considerare è **215**.

```
# Calcola la varianza spiegata cumulativa
total_explained_variance = pca.explained_variance_ratio_.cumsum()

# Calcola il numero di componenti principali necessarie per spiegare almeno il 95% della varianza totale
n_over_95 = len(total_explained_variance[total_explained_variance>=.95])

#Calcola il numero di componenti aggiuntive che sarebbero necessarie per raggiungere il 95% di varianza spiegata totale
n_to_reach_95 = encoded_tags_df.shape[1] - n_over_95
```

5. Data Modeling

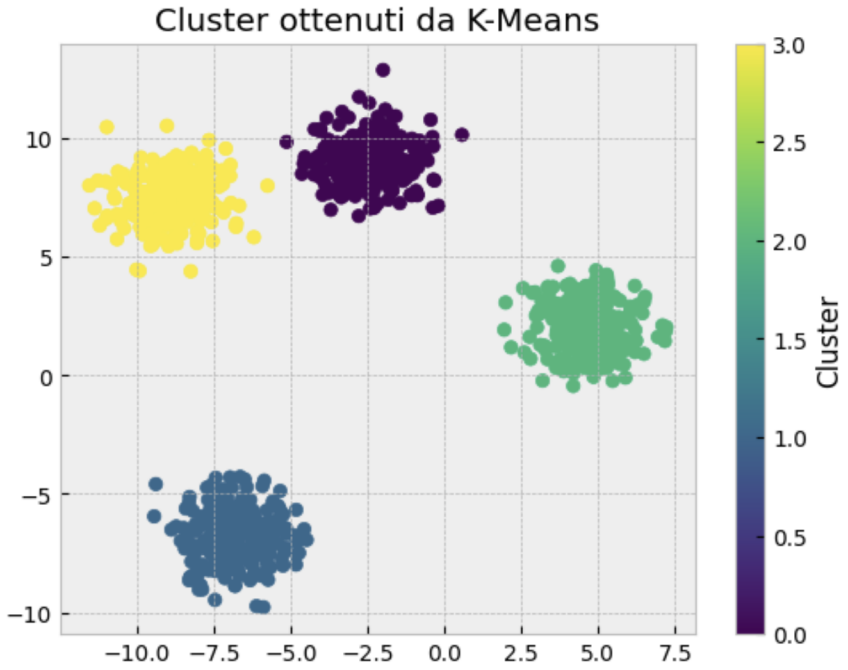
Nella fase di *Data Modeling* verranno determinate e valutate le tecniche finalizzate alla costruzione del modello, dopodiché ed si passerà alla fase di addestramento di quest'ultimo, durante la quale verranno configurati i parametri, verrà addestrato e si commenteranno i risultati ottenuti. Il problema affrontato è di *apprendimento non supervisionato*. La strategia utilizzata sarà basata sui cluster, che si adattano bene a questo genere di problemi.

5.1 K-Means

K-Means è uno degli algoritmi di clustering più popolari e utilizzati nell'ambito del Machine Learning non supervisionato. La sua funzione principale è quella di suddividere un insieme di dati in gruppi omogenei, chiamati cluster, in base alle somiglianze tra di essi. Per implementare il K-Means in Python abbiamo utilizzato la libreria *sklearn*.

Per utilizzare il K-Means bisogna tuttavia conoscere a priori il numero di cluster che verranno utilizzati. Abbiamo stimato che il numero di cluster ideali sia compreso tra 1 e 9. Quindi chiameremo il K-Means per ogni k da 1 a 9 e per ciascun k calcoleremo l'**inerzia**: definita come la somma delle distanze quadrate di ogni punto dati dal centroide del cluster più vicino, l'inerzia è una metrica utilizzata per valutare la coesione dei cluster. Indica quanto i punti dati in ciascun cluster sono vicini al centroide del loro rispettivo cluster.

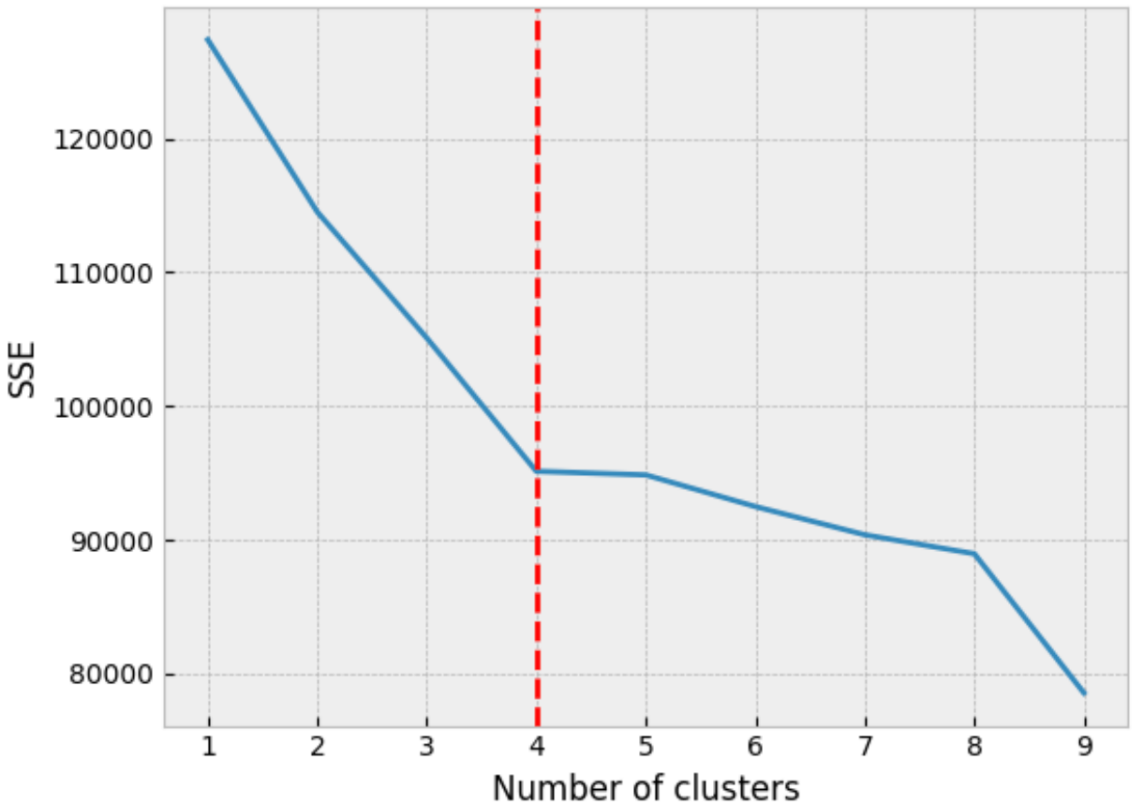
```
    sse = {}
    for k in range(1, 10):
        kmeans = KMeans(n_clusters=k, max_iter=1000, random_state=42).fit(X)
        X["clusters"] = kmeans.labels_
        sse[k] = kmeans.inertia_
```



Elbow Point

Il **Punto di Gomito** viene utilizzato per valutare se il numero di cluster utilizzati per suddividere i dati è ottimale e cioè se aggiungendo nuovi cluster non riscontriamo un significativo miglioramento della coesione all'interno dei cluster.

Possiamo notare dalla seguente immagine che abbiamo in particolare due punti di gomito significativi, cioè quando il numero di cluster è **4** oppure **8**.



5.1.1 Definizione del Modello e Addestramento

Individuati i numeri di cluster ottimali, cioè 4 e 8, vedremo quale suddivisione restituirà i risultati migliori.

Partizione del Dataset

Prima di procedere all'addestramento del modello, è necessario ripartire il dataset di partenza in *training set* e *test set*. In particolare abbiamo destinato il 20% del dataset per la creazione del test set, come mostrato dal seguente codice:

```
    # Sceglie la partizione del dataset da destinare al training set
    X_train, X_test = train_test_split(X, test_size=0.2, random_state=42)
```

Dove *random_state*=42 garantisce la riproducibilità della divisione, cioè otterremo la stessa divisione dei dati ogni volta che il codice viene eseguito con lo stesso seed.

Soluzione con 4 Cluster

Cominceremo considerando 4 cluster, procedendo per le seguenti fasi:

1. si definiscono i parametri del modello, tra cui il numero di cluster (in questo caso 4), il seed per garantire la riproducibilità dei risultati, e il numero massimo di iterazioni per il clustering;
2. si comincia con l'addestramento del modello partendo dai dati di test precedentemente definiti;
3. viene eseguita la **10-Fold Cross-Validation**[1] per per valutare le prestazioni del modello sui dati di test ed i risultati vengono salvati;
4. viene eseguita la predizione dei cluster per i dati di test ed i risultati vengono salvati.

```
    # Definizione dei parametri con cui l'oggetto KMeans sarà istanziato
    kmeans_4 = KMeans(n_clusters=4, random_state=42, max_iter=1000)
    # Addestramento del modello sui dati di test
    kmeans_4.fit(X_train)
    kmeans_4.fit_predict(X_train)
    # Calcola i punteggi della Cross-Validation
    scores = cross_val_score(kmeans, X_test, cv=10)
    # Salva i risultati della Cross-Validation
    pred = cross_val_predict(kmeans, X_test, cv=10)
```

Dall'esecuzione di questo codice risulta che lo score massimo calcolato per il modello a 4 cluster risulta essere di -2239.3616573250965.

Soluzione con 8 Cluster

Utilizzeremo un codice simile per un modello ad 8 cluster:

```
    # Definizione dei parametri con cui l'oggetto KMeans sarà istanziato
    kmeans_8 = KMeans(n_clusters=8, random_state=42, max_iter=1000)
    # Addestramento del modello sui dati di test
    kmeans_8.fit(X_train)
    kmeans_8.fit_predict(X_train)
    # Calcola i punteggi della Cross-Validation
    scores = cross_val_score(kmeans_8, X_test, cv=10)
    # Salva i risultati della Cross-Validation
    pred = cross_val_predict(kmeans_8, X_test,cv=10)
```

Dall'esecuzione di questo codice risulta che lo score massimo calcolato per il modello a 8 cluster risulta essere di -1575.9311996934612.

[1] Metodo statistico che consiste nella ripetuta partizione e valutazione dell'insieme dei dati di partenza. In particolare, nella 10-Fold i dati verranno divisi in 10 gruppi, di cui 1 per il test set e k-1 per il training set.

6. Evaluation

6.1 Scelta delle Metriche di Valutazione

Per la scelta del modello migliore ci siamo avvalsi in particolare di due metriche di valutazione:

- **Silhouette Score:** misura quanto ogni punto in un cluster è simile agli altri punti nello stesso cluster. Il valore del Silhouette Score varia da -1 a 1 e un punteggio più alto indica che il campione è stato clusterizzato in modo corretto. Un valore di Silhouette Score vicino a 1 indica che i campioni sono stati clusterizzati correttamente mentre un valore di Silhouette Score vicino a 0 indica che i campioni si trovano vicino ai confini tra due cluster. Infine, un valore negativo di Silhouette indica che un campione potrebbe essere stato assegnato al cluster sbagliato.
- **Davies-Bouldin Score:** misura la "distorsione" media tra i cluster. Minore è il valore, migliore è il clustering. Un valore più basso indica cluster più compatti e distinti tra loro. Il valore minimo di Davies-Bouldin Score è 0, che indica il clustering ideale.

6.2 Valutazione Modelli

Soluzione con 4 Cluster

```
    labels = kmeans_4.predict(X_test)
    # Calcolo Silhouette Score
    silhouette = silhouette_score(X_test,labels)
    # Calcolo Davies-Bouldin Score
    davies = davies_bouldin_score(X_test,labels)
```

L'esecuzione del codice precedente per il modello a 4 cluster ci offre i seguenti due risultati:

1. Silhouette Score: 0.37744084144674783
2. Davies-Bouldin Score: 1.0963214857490438

Soluzione con 8 Cluster

```
    labels = kmeans_8.predict(X_test)
    # Calcolo Silhouette Score
    silhouette = silhouette_score(X_test,labels)
    # Calcolo Davies-Bouldin Score
    davies = davies_bouldin_score(X_test,labels)
```

L'esecuzione del codice precedente per il modello a 8 cluster ci offre i seguenti due risultati:

1. Silhouette Score: 0.3520217797336636
2. Davies-Bouldin Score: 1.2827546914921293

6.3 Comparazione Modelli

Verranno in questa sezione confrontati e analizzati i risultati ottenuti precedentemente.

	Silhouette Score	Davies-Bouldin Score
4 Cluster	0.3774408414467478	1.0963214857490438
8 Cluster	0.3520217797336636	1.2827546914921293

Table 6.1: Tabella Riassuntiva

Silhouette Score. Per il modello a 4 cluster, il punteggio di Silhouette è di circa 0.377, indicando una buona separazione dei cluster, mentre per il modello a 8 cluster, il punteggio di Silhouette è leggermente inferiore, circa 0.352, indicando una separazione leggermente meno efficace rispetto al modello a 4 cluster.

Davies-Bouldin Score Per il modello a 4 cluster, il punteggio di Davies-Bouldin è di circa 1.096, il che è relativamente basso e suggerisce una buona dispersione dei cluster,mentre per il modello a 8 cluster, il punteggio di Davies-Bouldin è leggermente più alto, indicando una maggiore dispersione dei cluster rispetto al modello a 4 cluster.

In definitiva. Emerge dai dati raccolti che la soluzione con 4 cluster costituisca la scelta migliore.

7. Conclusioni

7.1 Integrazione con Kawaii-Comix

Per integrare il modulo con l'e-commerce Kawaii-Comix si è effettuato un collegamento con un server **Flask**[1]. Tale server riceve una richiesta asincrona da **AJAX**[2] attraverso uno script JavaScript, chiamando il sistema di raccomandazione Kawaii-Bot. Quest'ultimo a sua volta restituirà una lista di esattamente cinque prodotti con caratteristiche simili al prodotto su cui l'utente si trova. La lista elaborata non è statica, ma cambia quando la pagina viene aggiornata basandosi su una lista completa di prodotti simili.

7.2 Miglioramenti

Per il futuro può essere considerata anche l'integrazione del valore di rating per l'addestramento del modello, in quanto è indice dell'interesse dell'utenza e può risultare valido per la formulazione di predizioni.

[1] Flask è un web framework open source scritto con Python caratterizzato da flessibilità, leggerezza e semplicità d'uso. Fa parte della categoria dei micro-framework per via del suo approccio allo sviluppo minimalista non opinionato, che lascia agli sviluppatori la possibilità di scegliere quando e soprattutto come implementare ogni aspetto delle proprie web app a partire dalla struttura del progetto, scegliendo quali funzionalità è davvero necessario implementare, quanti e quali file/moduli creare, le convenzioni di sviluppo a cui fare riferimento è così via.

[2] AJAX consente di leggere dati da un server web dopo il caricamento della pagina, aggiornare una pagina web senza ricaricarla e inviare dati a un server web in background.